SOFTCOMPUTER CLUB ITALIA
RIVISTA No. 1 - del 02/06/87 - Socio No. 74A

BENVENUTI tra noi, Colgo l'occasione di ringraziarVi tutti, che con le vostre adesioni avete permesso la nascita di questo CLUB riservato ai posessori del TI-99/4A e/o GENEVE.

In tutti i nostri appuntamenti cercheremo di aiutarVi a conoscere di piu' il TI-99/4A, insegnandoVi tutto cio' che serve per diventare buoni programmatori ed utilizzatori, informandovi su tutto cio' che esiste ancora disponibile e cio' che sara' a disposizione in futuro.

Come noterete non esiste una copertina, ma questo e' stato scelto per lasciare piu' spazio alla rivista, alla quale anche Voi potrete parteciparvi con articoli, programmi, domande oppure solo informazioni.

In questo numero, tratteremo l'Assembler, le notizie dal mondo, le novita' le occasioni, in particolare modo l'argomento piu' discusso sara' come imparare l'Assembler, passo per passo.

Per qualsiasi problema potete scrivere al nostro indirizzo e' non appena possibile sara' pubblicata la risposta su questa rivista.

Vi chiediamo di non passare la rivista ad amici, ma al limite imprestarla in maniera che possa conoscerla ed apprezzarla cosi' da abbonarsi anche lui, il prezzo della stessa e' un po' cara, ma questo non e' dovuto a noi ma al costo delle fotocopie, che speriamo di abbassare con l'acquisto di una fotocopiatrice d'occasione.

Si ringrazia per gli aiuti e l'assistenza:

ENNIO MEDICI, ROBERTO COMPARE, MAURA BOERI, CHICAGO USER GROUP, RYTE DATA, TEXAS INSTRUMENTS, MILLERS GRAPHIC, ed altri che in un modo o nell'altro hanno collaborato anche se non direttamente alla riuscita di questa pubblicazione.

Da questo primo numero noterete dei fogli all'interno con la titolazione

== TABELLA DI COMPARAZIONE.....==

questi fogli dovranno essere tenuti via con tutte le cosiderazioni, saranno utilissimi nella comprensione e stesura dei programmi che pubblicheremo o che vorrete creare Voi stessi.

L'Assembler (da ora in poi E/A) puo' sembrare difficile per chi non si' e' interessato piu' di quel tanto per impararlo, certamente piu' complicato del Basic, infatti quest'ultimo ha gia' istruzioni, precedentemente inserite dal costruttore, che ti permettono un uso immediato di procedure per costruire i tuoi programmi.

Ma questi non sono direttamente comprensibili dal Computer, infatti hanno bisogno di essere tradotti in un linguaggio piu' vicino a come lavora il TI-99/4A, ed ecco che occorre un INTERPRETE BASIC che serve proprio a questo, ed ecco perche' il nostro gioiello in Basic e molto lento.

L'E/A invece non ti permette un uso immediato dell'istruzione ma un uso indiretto tramite un Assemblatore che ti compila il tuo programma in un formato comprensibile dal computer.

In questa maniera il tuo programma sara' direttamente compreso dalla macchina e girera' in una velocita' che a volte deve essere rallentata per non perdere il controllo di esso.

Per capire meglio noi spiegheremo passo per passo ogni istruzione usata e metteremo al confronto il basic.

Innanzitutto dovete capire il significato dei Registri; sono speciali Word 16 Bit o due Bytes indicati per memorizzare ed usare, manipolare i dati contenuti in essi, e per fare questo dovete almeno avere un po' di pratica con la numerazione binaria ed esadecimale.

Nel caso contrario potrete usare un programma basic incluso nei dischetti con il nome di CONVERTER, esso automaticamente trasforma tutti i vostri dati nella forma che volete. (DISCHI disponibili solo per i Soci)

Ci sono 16 Registri generali chiamati WORKSPACE (spazio di lavoro) accessibili al programmatore, e sono numerati da 0 a 15, i registri sono l'area di lavoro della CPU (unita' di elaborazione centrale), essi sono usati per l'indirizzamento, il calcolo e la manipolazione dei bits.

Svolgono compiti che altre aree del computer non possono fare, essi occupano sempre un' area di memoria contigua per un totale di 32 Bytes (HEX >20), ricordatevi che voi potete designare nel vostro programma altri 32 Bytes, per essere usati come spazio di lavoro, ma solo un set di registri per volta puo' essere usato.

Vi sono altri 3 Registri (HARDWARE) usati dal computer, e sono P.C., W.P., e ST., ogniuno di questi ha una funzione ben precisa:

P.C. - (PROGRAM COUNTER) i valori di questo Registro sono usati in unione con altri Registri dati per localizzare la prossima istruzione da eseguire nel vostro programma, in pratica come in Basic, se non si interviene con un istruzione di salto il computer procedera' come sono inserite le istruzioni.

Il P.C., come le istruzioni del programma vengono eseguite, viene incrementato puntando all'indirizzo della prossima istruzione logica, a meno che non si intervenga su di esso per indirizzare il computer verso un altra istruzione.

PROGRAM COUNTER REGISTER (P.C.).

```
HARDWARE                                ISTRUZIONI PROGRAMMI

REGISTRI                              RESIDENTI IN MEMORIA

TMS 9900                                   MEMORIA

+----------+                        +-------------------------+
! +------+ !                        ! ISTRUZIONE No.1         !
! !P.C.!--------------\             +-------------------------+
! +------+ !             \          ! ISTRUZIONE No.2         !
!          !              \         +-------------------------+
! +------+ !               \------>! ISTRUZIONE No.3         !
! !W.P.! !                           +-------------------------+
! +------+ !                        ! ISTRUZIONE No.4         !
!          !                        +-------------------------+
! +------+ !                        ! ISTRUZIONE No.5         !
! !ST. ! !                           +-------------------------+
! +------+ !                        ! Etc. Etc. Etc.          !
+----------+                        +-------------------------+
```

Il P.C. dopo ogni istruzione si autoincrementa per eseguire in maniera sequenziale tutte le istruzioni inserite nel programma.

W.P. - (WORKSPACE) i valori contenuti in esso servono a contenere l'indirizo dello spazio di lavoro prossimo, puntando all'indirizzo iniziale dei 16 Registri generali. Possono essere definiti molti gruppi di Registri WorkSpace, e ogni gruppo puo' essere accessibile dalla manipolazione del programma dagli indirizzi contenuti in questi Registri.

WORKSPACE POINTER REGISTER (W.P.)

```
 HARDWARE                          SET ISTRUZIONI PROGRAMMA

 REGISTRI                     E SPAZIO DI LAVORO (W.P.) REGISTRI

 TMS 9900                                 MEMORIA
+----------+                        +-----------------------+
! +-----+  !                        !                       !
! !P.C.!   !                        !                       !
! +-----+  !                        !  PROGRAMMA            !
!          !                        !                       !
! +-----+  !                        !                       !
! !W.P.!---------\                  +-----------------------+
! +-----+  !      \                 !  WorkSpace  "A"       !
!          !       \                !                       !
! +-----+  !        ---------------! R1 R2 R3 R4 R5 R6     !
! !ST.  !  !                        !                       !
! +-----+  !                        ! R7 R8 R9 R10 R11      !
+----------+                        !                       !
                                    ! R12 R13 R14 R15       !
                                    +-----------------------+
                                    !  WorkSpace  "B"       !
                                    !                       !
                                    ! R1 R2 R3 R4 R5 R6     !
                                    !                       !
                                    ! R7 R8 R9 R10 R11      !
                                    !                       !
                                    ! R12 R13 R14 R15       !
                                    +-----------------------+
```

ST. - (STATUS REGISTER) i valori contenuti in questo Registro contengono lo stato del computer dell'ultima istuzione eseguita, i bit individuali a seconda di certe condizioni sono azzerati o fissati ad uno.

Si useranno molto spesso, specie nei calcoli ed in altre situazioni, ad esempio volete confrontare il valore di A e di B,immediatamente dopo che il computer ha eseguito l'istruzione di confronto il Registro di stato indichera' alcuni valori che debitamente interpretati vi porteranno soluzione del problema.

REGISTRO DI STATO (ST.)

POSIZIONE BIT		SIGNIFICATO SE FISSATO AD "1"
0	L>	LOGICAMENTE MAGGIORE DI.....
1	A>	ARITMETICAMENTE MAGGIORE DI.
2	EQ	UGUAGLIANZA.
3	C	RIPORTO.
4	OV	TRABOCCAMENTO (OWERFLOV).
5	OP	PARITA' DISPARI.
6	X	OPERAZIONE ESTESA.
07/11	-	NON USATI.
12/15	-	MASCHERA DI INTERRUZIONE.

Il Registro di stato puo' permettervi di conoscere se due valori sono uguali, o dare evidenza a molte altre condizioni, in ogni caso le operazioni aritmetiche trattano tutti i Bytes come rappresentanti un valore aritmetico, mentre le operazioni logiche trattano i Bytes come rappresentanti una serie di Bit.

Varie istruzioni in E/A vi permetteranno di istruire il computer in maniera di interpretare come voi volete i valori contenuti di un Byte.

Per un migliore apprendimento dei Registri di Stato abbiamo stampato una piccola tabella di comparazione, in questa maniera potrete avere un confronto immediato di tutte le istruzioni che influiscono su di essi ed esaminare quali sono stati influiti e quali no.

In ogni caso la stessa non puo' essere completa come il manuale E/A, infatti una tabella completa comporterebbe la traduzione e stampa completa di tutto il manuale, che e' gia' disponibile tradotto in Italiano presso Ennio Medici, il suo indirizzo lo troverete piu' avanti leggendo la rivista.

La tabella in questione potra' esservi utile nella decodificazione dei nostri e vostri programmi, se per esempio in un programma trovate le seguenti istruzioni, leggendo la tabella troverete il significato di cio' che si e' fatto:

Ad esempio:

```
A     BIC,R1
CI    R1,676
JEQ   X
```

A (Addizziona Word) addiziona il valore contenuto in **BIC** con R1 ed il risultato viene messo in R1.

CI confronta immediatamente il valore di R1 con 676, ora leggete quali bit sono influiti nel confronto con CI nella tabella, e troverete questo:

	L>	A>	EQ	C	O	OP	X
EQ	I	I	I	N	N	N	N

Cio' indica che i bit influiti (I) sono:

L> Logicamente Maggiore di...

A> Aritmeticamente Maggiore di.....

EQ Uguale....

Se nel confronto (CI) R1 era uguale a 676 il Bit di Stato sarebbe stato settato a 1, in posizione EQ all'incirca in questa maniera

0010000000000000

Se invece nel confronto R1 era maggiore di 676 il Bit di Stato sarebbe stato settato a 1, in posizione A> all'incirca cosi':

0100000000000000

L'istruzione JEQ legge il Bit di Stato nella posizione EQ e se il valore incontrato risulta essere 1 salta alla istruzione X altrimenti prosegue.

Nel caso si fosse adoperato JGT allora il Bit di Stato esaminato sarebbe stato A> e se fosse stato settato a 1 saltava alla istruzione X altrimenti proseguiva.

In teoria si procede come le istruzioni in Basic IF THEN ELSE, chiaramente la semplicita' di questo linguaggio la si deve scordare ma in compenso in E/A si possono ottenere risultati che altrimenti sarebbero impossibili.

Non abbiate paura di programmare in questo linguaggio, basta solo un po' di impegno e di pratica per riuscire a fare gia' qualcosa di discreto, le norme per la programmazione sono identiche per tutti i linguaggi, bisogna avere le idee chiare di cosa si vuole fare, quindi schizzare su un foglio gia' le vostre subroutine, e fare un elenco di tutte le variabili che userete.

Noi per insegnarvi meglio vi faremo moltissimi esempi ben documentati e gia' subito assemblabili per un uso e controllo pratico piu' veloce, inoltre per una facile comprensione, metteremo una istruzione Basic con accanto un istruzione Basic simile all'E/A, e l'istruzione E/A vera e propria in fondo.

Noi procederemo in questa maniera, per esempio volete stampare il vostro nome alla riga 12 e colonna 10:

In Basic fareste cosi': - In Basic piu' simile all'Assembler cosi':

```
100 CALL CLEAR                  -
110 DISPLAY AT(12,10):"NOME"-
120 GOTO 120                    -
130 END                         -
```

```
- 100 A$="NOME"
- 110 B=4 ! LUNGHEZZA DEL NOME
- 120 C=32 ! COD. DELLO SPAZIO
- 130 CALL HCHAR(1,1,C,768)
- 140 DISPLAY AT(12,10)SIZE(B)
-      :A$
- 150 GOTO 150
- 160 END
```

Per comodita' abbiamo stampato a parte il programma in E/A in maniera di poter allegare in un unico fascicolo tutti i nostri esempi cosi' da poter avere una facile consultazione per usi futuri.

Come noterete la lunghezza in un programma Assembler e superiore a quello del Basic, ma questo e' dovuto anche all'obbligo di elencare tutte le matrici che userete e di assegnare a loro uno spazio di memoria su cui lavoreranno, in Basic l'assegnazione di una stringa puo' essere effettuata al momento dell'uso stesso, ma in E/A no.

Il programma allegato ha gia' una sua spiegazione riga per riga, in ogni caso Noi Vi consigliamo l'acquisto del manuale E/A in Italiano per una consultazione immediata delle istruzioni usate, il manuale in questione lo potrete trovare da:

ENNIO MEDICI
VIA STAZIONE 21
18030 BEVERA (IM)
TEL. 0184/210068 (ORE PASTI)

Altri libri sono disponibili sempre dalla stessa persona, per imparare l'Assembler e sono gia' tradotti in Italiano, per ogni vostro bisogno chiedete a Ennio Medici, lui sara' ben lieto di aiutarvi nella vostra scelta.

Il programma allegato viene presentato nel formato sorgente e nel formato stampato dall'Assemblatore, questo viene fatto per una migliore comprensione dello stesso da parte vostra.

Mettete a confronto i due listati noterete che i numeri di linea (PRESENTI nella stampa del listato sorgente per motivi di testo) corrispondenti alle istruzioni e le stesse non sono cambiate, ma in compenso tra loro apparsi dei numeri elencati in due colonne, per procedere in ordine i numeri vanno decifrati in questa maniera:

Il primo e' il numero di linea,

Il secondo e' il contatore di locazione, con l'E/A inizia da >0000 ed e' incrementato linea per linea di 2,

Il terzo numero e' la rappresentazione in esadecimale (Hex) del codice macchina di ogni indirizzo (Valore del contatore di locazione). Per esempio alla linea 5 locazione >0000 voi vedrete il valore >0020 decimale 32 corrispondente al carattere spazio.

Tutto questo oltre ad un utile supporto per la correzione dei propri errori serve anche per sapere quanto e' lungo il vostro programma, e per fare cio' basta sottrarre il valore iniziale del contatore di locazione con quello finale, nel nostro caso 0064-0000 = 64 cio' indica che il programma e' lungo 64 Bytes.

Se ci fossero dei dubbi o non avete capito qualche cosa non vergognateVi di telefonare o scrivere, noi siamo qui' per questo.

Ad esempio il nostro collega che tiene la contabilita' del Club non aveva capito il perche' bisognava muovere il valore contenuto in R11 nella etichetta **RT**, nella linea 0010, noi per favorire tutti abbiamo pensato di pubblicare la risposta.

Quando noi chiamiamo un programma in Assembler al momento della chiamata il Registro R11 contiene il valore della locazione successiva a quella della chiamata, quindi noi una volta caricato il programma e messo in funzione dobbiamo salvare questo valore per poterci tornare una volta eseguito lo stesso.

In pratica se noi partiamo dall'opzione 3 dell'E/A noi torneremo ad essa se abbiamo strutturato il programma per fare questo, nel caso di una chiamata da X/B o da TI/Basic durante il programma noi torneremo alla istruzione successiva della chiamata e messa in funzione della routine in E/A.

In questo nostro primo incontro abbiamo cercato di iniziare mettendo Voi in grado di poterci seguire senza molti sforzi, per fare questo abbiamo dovuto aggiungere al normale numero di pagine (15) quelle necessarie al completamento del supporto tecnico, con anche la sacrificazione di alcune pagine di articoli vari.

Per non chiudere solo con un articolo vi accenniamo una piccola parentesi di novità dall'estero, infatti ci ripromettiamo di ampliare e completare meglio questo spazio.

Dal Canada: Ryte Data
210 Mountain Street
Haliburton, Ontario
K0M 1S0, CANADA

Ha inserito nel suo catalogo un Kit che permette di modificare il sistema operativo del Disk Controller Card PHP 1240 della Texas, in maniera di ottenere un controller capace di Doppia Faccia, Doppia Densita o Quadrupla Densita (80 tracce), con una spesa di 45 $, aggiungendoci 29.95 $ potrete avere anche disk manager su disco a load automatico.

Per chi non fosse in grado di operare saldature sulla scheda per scarsa capacita' puo' rivolgersi a noi, inoltre possiamo anche a provvedere all'acquisto del Kit per conto vostro previo accordi telefonici ed all'invio dell'importo in DOLLARI, aggiungendo le spese di spedizione e quelle di sdoganamento.

Dalla Ditta: BUD MILLS SERVICES
166 DARTMOUTH DR.
TOLEDO OHIO 43614

E' disponibile un Kit che inserito nella consolle del TI-99/4A, permette allo stesso di essere [illegible] **: L 50 (per cento)** piu' veloce del normale, per ora non sono ancora disponibili notizie piu' precise, ma dal depliants inviato si nota l'inserimento di 64Kram all'interno di essa e di due Chips, il costo del materiale escluso le spese di spedizione si aggira sui 58 $.

Per ulteriori notizie l'appuntamento e' nel prossimo numero.

Nel prossimo numero tratteremo:

Il colore,
I numeri e le operazioni,
Le novita' dagli U.S.A.
Le occasioni.

Per ora e' tutto l'appuntamento e per il prossimo mese, nel frattempo scrivete, scrivete, scrivete, e sopratutto difondete la notizia.

Arrivederci a presto con il TI-99/4A (L'IMMORTALE)

TABELLA DI COMPARAZIONE PER CARATTERI E COLORI

XB-TI DECIMALE	E/A ESADEC.	E/A SPIAZZ.	CARATTERE	TI-XB SET	E/A SET	
0	00	00	NUL	-	00	
1	01	08	SOH	-	00	
2	02	10	STX	-	00	
3	03	18	ETX	-	00	
4	04	20	EOT	-	00	
5	05	28	ENQ	-	00	
6	06	30	ACK	-	00	
7	07	38	BEL	-	00	
8	08	40	BS	-	01	
9	09	48	HT	-	01	
10	0A	50	LF	-	01	
11	0B	58	VT	-	01	
12	0C	60	FF	-	01	
13	0D	68	CR	-	01	
14	0E	70	SO	-	01	
15	0F	78	SI	-	01	
16	10	80	DLE	-	02	
17	11	88	DC1	-	02	
18	12	90	DC2	-	02	
19	13	98	DC3	-	02	
20	14	A0	DC4	-	02	
21	15	A8	NAK	-	02	
22	16	B0	SYN	-	02	
23	17	B8	ETB	-	02	
24	18	C0	CAN	-	03	
25	19	C8	EM	-	03	
26	1A	D0	SUB	-	03	
27	1B	D8	ESC	-	03	
28	1C	E0	FS	-	03	
29	1D	E8	GS	-	03	
30	1E	F0	RS	0	03	(CURSORE ESCLUSO TI)
31	1F	F8	US	0	03	(ESCLUSO TI)
32	20	100	SPAZIO	1	04	
33	21	108	! P. ESC.	1	04	
34	22	110	" VIRG.TE	1	04	
35	23	118	# LIBBRA	1	04	
36	24	120	$ DOLLARO	1	04	
37	25	128	% PER 100	1	04	
38	26	130	& AND	1	04	
39	27	138	' APOSTR.	1	04	

TABELLA DI COMPARAZIONE PER CARATTERI E COLORI

XB-TI DECIMALE	E/A ESADEC.	E/A SPIAZZ.	CARATTERE	TI-XB SET	E/A SET
40	28	140	(PAR. A.	2	05
41	29	148	) PAR. C.	2	05
42	2A	150	* ASTER.	2	05
43	2B	158	+ PIU'	2	05
44	2C	160	, VIRGOLA	2	05
45	2D	168	- MENO	2	05
46	2E	170	. PUNTO	2	05
47	2F	178	/ BARRA O	2	05
48	30	180	0 ZERO	3	06
49	31	188	1 UNO	3	06
50	32	190	2 DUE	3	06
51	33	198	3 TRE	3	06
52	34	1A0	4 QUATTRO	3	06
53	35	1A8	5 CINQUE	3	06
54	36	1B0	6 SEI	3	06
55	37	1B8	7 SETTE	3	06
56	38	1C0	8 OTTO	4	07
57	39	1C8	9 NOVE	4	07
58	3A	1D0	: DUE P.	4	07
59	3B	1D8	; P. E V.	4	07
60	3C	1E0	< MINORE	4	07
61	3D	1E8	= UGUALE	4	07
62	3E	1F0	> MAGG.	4	07
63	3F	1F8	? P. INT.	4	07
64	40	200	@ AT	5	08
65	41	208	A	5	08
66	42	210	B	5	08
67	43	218	C	5	08
68	44	220	D	5	08
69	45	228	E	5	08
70	46	230	F	5	08
71	47	238	G	5	08
72	48	240	H	6	09
73	49	248	I	6	09
74	4A	250	J	6	09
75	4B	258	K	6	09
76	4C	260	L	6	09
77	4D	268	M	6	09
78	4E	270	N	6	09
79	4F	278	O	6	09

TABELLA DI COMPARAZIONE PER CARATTERI E COLORI

XB-TI DECIMALE	E/A ESADEC.	E/A SPIAZZ.	CARATTERE	TI-XB SET	E/A SET
80	50	280	P	7	0A
81	51	288	Q	7	0A
82	52	290	R	7	0A
83	53	298	S	7	0A
84	54	2A0	T	7	0A
85	55	2A8	U	7	0A
86	56	2B0	V	7	0A
87	57	2B8	W	7	0A
88	58	2C0	X	8	0B
89	59	2C8	Y	8	0B
90	5A	2D0	Z	8	0B
91	5B	2D8	[P. Q. A	8	0B
92	5C	2E0	\ B. D. I	8	0B
93	5D	2E8	] P. Q. C	8	0B
94	5E	2F0	POTENZA	8	0B
95	5F	2F8	_ TRATTO	8	0B
96	60	300	` ACC. G.	9	0C
97	61	308	a	9	0C
98	62	310	b	9	0C
99	63	318	c	9	0C
100	64	320	d	9	0C
101	65	328	e	9	0C
102	66	330	f	9	0C
103	67	338	g	9	0C
104	68	340	h	10	0D
105	69	348	i	10	0D
106	6A	350	j	10	0D
107	6B	358	k	10	0D
108	6C	360	l	10	0D
109	6D	368	m	10	0D
110	6E	370	n	10	0D
111	6F	378	o	10	0D
112	70	380	p	11	0E
113	71	388	q	11	0E
114	72	390	r	11	0E
115	73	398	s	11	0E
116	74	3A0	t	11	0E
117	75	3A8	u	11	0E
118	76	3B0	v	11	0E
119	77	3B8	w	11	0E

TABELLA DI COMPARAZIONE PER CARATTERI E COLORI

XB-TI DECIMALE	E/A ESADEC.	E/A SPIAZZ.	CARATTERE	TI-XB SET	E/A SET	
120	78	3C0	x	12	0F	
121	79	3C8	y	12	0F	
122	7A	3D0	z	12	0F	
123	7B	3D8	{ P. G. A	12	0F	
124	7C	3E0	| L. SEP.	12	0F	
125	7D	3E8	} P. G. C	12	0F	
126	7E	3F0	~ TILDE	12	0F	
127	7F	3F8	DEL	12	0F	(DEL COME SPAZIO)
128	80	400		13	10	(I CARATTERI CHE)
129	81	408		13	10	(SEGUONO SONO)
130	82	410		13	10	(DISPONIBILI COME)
131	83	418		13	10	(SPAZIO)
132	84	420		13	10	
133	85	428		13	10	
134	86	430		13	10	
135	87	438		13	10	
136	88	440		14	11	
137	89	448		14	11	
138	8A	450		14	11	
139	8B	458		14	11	
140	8C	460		14	11	
141	8D	468		14	11	
142	8E	470		14	11	
143	8F	478		14	11	
144	90	480		15	12	(I CARATTERI CHE)
145	91	488		15	12	(SEGUONO NON SONO)
146	92	490		15	12	(DISPONIBILI IN)
147	93	498		15	12	(EXTENDED BASIC)
148	94	4A0		15	12	
149	95	4A8		15	12	
150	96	4B0		15	12	
151	97	4B8		15	12	
152	98	4C0		16	13	
153	99	4C8		16	13	
154	9A	4D0		16	13	
155	9B	4D8		16	13	
156	9C	4E0		16	13	
157	9D	4E8		16	13	
158	9E	4F0		16	13	
159	9F	4F8		16	13	

TABELLA DI COMPARAZIONE PER CARATTERI E COLORI

XB-TI DECIMALE	E/A ESADEC.	E/A SPIAZZ.	CARATTERE	TI-XB SET	E/A SET
160/168	A0/A7	500/540			14
169/176	A8/AF	548/580			15
177/184	B0/B7	588/5C0			16
185/192	B8/BF	5C8/600			17
193/200	C0/C7	608/640			18
201/208	C8/CF	648/680			19
209/216	D0/D7	688/6C0			1A
217/224	D8/DF	6C8/700			1B
225/232	E0/E7	708/740			1C
233/240	E8/EF	748/780			1D
241/248	F0/F7	788/7C0			1E
249/256	F8/FF	7C8/800			1F

I CARATTERI SOPRACITATI SI POSSONO USARE ESCLUSIVAMENTE SOLO IN ASSEMBLER PER MOTIVI DI SPAZIO NON SI E' PROVVEDUTO AD ELENCARE DETTAGLIATAMENTE I CODICI INFLUITI MA CON POCA DIFFICOLTA' CON I RIFERIMENTI INDICATI SARETE IN GRADO DI TROVARE CIO' CHE CERCATE, BUON LAVORO.....

TABELLA DI COMPARAZIONE DEL FORMATO SCHERMO TRA E/A E BASIC 32/c. 28/c.

C O L O N N E

```
FORMATO BASIC PRINT   28/c !            1         2          !
                           !  1234567890123456789012345678    !
                           !                                   !
FORMATO BASIC SCHERMO 32/c !          1         2         3  !
                           !1234567890123456789012345678901 2!
                           !                                   !
FORMATO  E/A  SCHERMO 32/c !          1         2         3  !
                           !0123456789012345678901234567890 1!
-------------------------------------------------------------!
            - 1 !      - 0  !LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!31
            - 2 !      - 32 !LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!63
        R   - 3 !      - 64 !LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!95
            - 4 !      - 96 !LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!127
        I   - 5 !  R   - 128!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!159
            - 6 !      - 160!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!191
        G   - 7 !  I   - 192!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!223
            - 8 !      - 224!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!255
        H   - 9 !  S   - 256!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!287
            - 10!      - 288!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!319
        E   - 11!  H   - 320!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!351
            - 12!      - 352!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!383
            - 13!  E   - 384!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!415
        B   - 14!      - 416!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!447
            - 15!      - 448!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!479
        A   - 16!  E   - 480!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!511
            - 17!      - 512!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!543
        S   - 18!  /   - 544!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!575
            - 19!      - 576!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!607
        T   - 20!  A   - 608!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!639
            - 21!      - 640!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!671
        O   - 22!      - 672!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!703
            - 23!      - 704!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!735
            - 24!      - 736!LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL!767
                             ----------------------------------
```

Per trovare il corrispondente E/A (riga e colonna) del basic procedere come segue:

1) Scegliere il formato corrispondente a quello richiesto se 28c o 32c
2) Con una riga leggere il formato corrispondente alla colonna
3) Scegliere il corrispondente tra riga Basic e riga E/A
4) Sommare i due valori ottenuti.

Es.) In Basic Colonna 12, Riga 9

Corrispondente Colonna 12 (Basic) = Colonna 11 (E/A)

Corrispondente Riga 9 (Basic) = Riga 256 (E/A)

Posizione corrispondente in Assembler = 267 (256+11)

TABELLA DI COMPARAZIONE PER REGISTRI E ISTRUZIONI

Bit di Stato esaminati dalle istruzioni di salto elencati in questi fogli.

MNEMONICO	L>	A>	EQ	C	OV	OP	Salta se:
JH	I	N	I	N	N	N	L>=1 e EQ=0
JL	I	N	I	N	N	N	L>=0 e EQ=0
JHE	I	N	I	N	N	N	L>=1 o EQ=1
JLE	I	N	I	N	N	N	L>=0 o EQ=1
JGT	N	I	N	N	N	N	A>=1
JLT	N	I	I	N	N	N	A>=0 e EQ=0
JEQ	N	N	I	N	N	N	EQ=1
JNE	N	N	I	N	N	N	EQ=0
JOC	N	N	N	I	N	N	C=1
JNC	N	N	N	I	N	N	C=0
JNO	N	N	N	N	I	N	OV=0
JOP	N	N	N	N	N	I	OP=1

TABELLA DI COMPARAZIONE PER REGISTRI E ISTRUZIONI

Registri di Stato influenzati dalle operazioni elencate in questi fogli.

NOME	NUMERO	SIGNIFICATO
L>	0	Logicamente Maggiore di.....
A>	1	Aritmeticamente Maggiore di..
EQ	2	UGUALE.....
C	3	Riporto.....
OV	4	Traboccamento.....
OP	5	Parita' o Disparita'.....
X	6	Estensione Operativa.....
-	7/11	Riservati.....
-	12/15	Maschera di Interruzione.....

TABELLA DI COMPARAZIONE PER REGISTRI E ISTRUZIONI

ISTRUZIONE - SIGNIFICATO -

A - Addizione di Word

Esempio :

A VAL,R2 * Addiziona il valore contenuto in VAL con R2 e mette il risultato in R2

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK  !
+-------------------------------------------------------+
 I     I  I  I  I  N  N
```

ISTRUZIONE - SIGNIFICATO -

AB - Addizione di BYTE

Esempio :

AB R2,R3 * Addiziona il Byte sinistro di R2 al Byte sinistro di R3 e mette il risultato in R3

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK  !
+-------------------------------------------------------+
 I  I  I  I  I  I  N
```

ISTRUZIONE - SIGNIFICATO -

ABS - Valore Assoluto

Esempio :

ABS *R2 Sostituisce il contenuto della Word che inizia allo indirizzo in R2 con il suo valore assoluto.

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-----------------!  INT. MASK    !
+-------------------------------------------------------+
 I  I  I  N  I  N  N
```

ISTRUZIONE - SIGNIFICATO -

AI - Addizione Immediata

Esempio :

AI R2,9 * Addiziona 9 ad R2 e mette il risultato in R2

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-----------------!  INT. MASK    !
+-------------------------------------------------------+
 I  I  I  I  I  N  N
```

ISTRUZIONE - SIGNIFICATO -

DEC - Decrementa di 1

Esempio :

DEC R2 * Decrementa di uno il valore contenuto in R2 ed il risultato viene messo in R2

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK  !
+-------------------------------------------------------+
 I  I  I  I  I  N  N
```

ISTRUZIONE - SIGNIFICATO -

DECT - Decrementa di 2

Esempio :

DECT R1 * Decrementa di 2 il valore contenuto in R1 e mette il risultato in R1.

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK  !
+-------------------------------------------------------+
 I  I  I  I  I  N  N
```

ISTRUZIONE - SIGNIFICATO -

DIV - Divisione

Esempio :

DIV **VAL,R3** * Divide il valore contenuto in VAL con R3 e mette il risultato in R3 e se c'e' il resto in R4.

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK   !
+-------------------------------------------------------+
 N  N  N  N  I  N  N
```

ISTRUZIONE - SIGNIFICATO -

INC - Incrementa di 1

Esempio :

INC R7 * Incrementa di 1 il valore contenuto in R7 e mette il risultato in R7

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK   !
+-------------------------------------------------------+
 I  I  I  I  I  N  N
```

ISTRUZIONE - SIGNIFICATO -

INCT - Incrementa di 2

Esempio :

INCT R5 * Incrementa di 2 il valore contenuto in R5 e mette il risultato in R5

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-----------------!  INT. MASK    !
+-------------------------------------------------------+
 I  I  I  I  I  N  N
```

ISTRUZIONE - SIGNIFICATO -

MPY - Moltiplicazione

Esempio :

MPY **VAL,R2** * Moltiplica il valore contenuto in R3 con il valore contenuto in VAL, il risultato viene messo in R3 e R4, cioe' in 32 Bit.

Bit di Stato influenzati Nessuno

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-----------------!  INT. MASK    !
+-------------------------------------------------------+
 N  N  N  N  N  N  N
```

ISTRUZIONE - SIGNIFICATO -

NEG - Negazione

Esempio :

NEG R2 * Sostituisce il contenuto di R2 con la somma del suo inverso in pratica sostituisce l'operando con lo stesso operando in complemento a due.

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK  !
+-------------------------------------------------------+
 I  I  I  N  I  N  N
```

ISTRUZIONE - SIGNIFICATO -

S - Sottrazione di Word

Esempio :

S R2,R3 * Sottrae il valore contenuto in R2 dal valore contenuto in R3 e mette il risultato in R3.

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK  !
+-------------------------------------------------------+
 1  I  I  I  I  N  N
```

TABELLA DI COMPARAZIONE PER REGISTRI E ISTRUZIONI

ISTRUZIONE	SIGNIFICATO
CB	Confronto di Byte

Esempio :

CB R2,R3 * Confronta i Byte di sinistra di R2 con i Byte di sinistra R3.

Bit di Stato influenzati

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L>	A>	EQ	C	OV	OP	X						INT. MASK			
I	I	I	N	N	I	N									

ISTRUZIONE	SIGNIFICATO
CI	Confronta Immediatamente

Esempio :

CI R1,767 * Confronta immediatamente il valore contenuto in R1 con 767.

Bit di Stato influenzati

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L>	A>	EQ	C	OV	OP	X						INT. MASK			
I	I	I	N	N	N	N									

ISTRUZIONE - SIGNIFICATO -

SB - Sottrazione di Byte

Esempio :

SB R2,R3 * Sottrae il Byte di sinistra di R2 dal Byte di sinistra di R3 e mette il risultato in R3

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK  !
+-------------------------------------------------------+
 I  I  I  I  I  I  N
```

ISTRUZIONE - SIGNIFICATO -

C - Confronto di Word

Esempio :

C R2,R3 * Confronta i contenuti di R2 con R3

Bit di Stato influenzati

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
+-------------------------------------------------------+
!L>!A>!EQ!C !OV!OP!X !-------------------!  INT. MASK  !
+-------------------------------------------------------+
 I  I  I  N  N  N  N
```

TABELLA DI COMPARAZIONE PER REGISTRI E ISTRUZIONI

ISTRUZIONE - SIGNIFICATO -

COC - Confronto di Bit corrispondenti a 1 (uno)

Esempio :

COC **VAL,R2** * Confronta i Bit contenuti in R2 con i Bit contenuti in VAL

Bit di Stato influenzati

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L>	A>	EQ	C	OV	OP	X						INT. MASK			
N	N	I	N	N	N	N									

ISTRUZIONE - SIGNIFICATO -

CZC - Confronto di Bit corrispondenti a 0 (zero)

Esempio :

CZC **VAL,R2** * Confronta ii bit contenuti in R2 con i Bit contenuti in VAL

Bit di Stato influenzati

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L>	A>	EQ	C	OV	OP	X						INT. MASK			
N	N	I	N	N	N	N									

```
0001                        DEF   START
0002                        REF   VSBW,VMBW
0003        837C   STATUS   EQU   >837C
0004 0000   2000   D        DATA  >2000
0005 0002   0000   RT       DATA  >0000
0006 0004     4E   A        TEXT  'NOME'
     0005     4F
     0006     4D
     0007     45
0007 0008          C        BSS   >20
0008 0028   C80B   START    MOV   R11,@RT
     002A   0002'
0009 002C   02E0            LWPI  C
     002E   0008'
0010 0030   06A0            BL    @CLEAR
     0032   0050'
0011 0034   0200   DISP     LI    R0,299
     0036   012B
0012 0038   0201            LI    R1,A
     003A   0004'
0013 003C   0202            LI    R2,4
     003E   0004
0014 0040   0420            BLWP  @VMBW
     0042   0000
0015 0044   C2E0   STOP     MOV   @RT,R11
     0046   0002'
0016 0048   04E0            CLR   @STATUS
     004A   837C
0017 004C   064B            DECT  R11
0018 004E   045B            RT
0019 0050   04C0   CLEAR    CLR   R0
0020 0052   0201            LI    R1,D
     0054   0000
0021 0056   0420   LOOP     BLWP  @VSBW
     0058   0000
0022 005A   0280            CI    R0,767
     005C   02FF
0023 005E   1302            JEQ   RET
0024 0060   0580            INC   R0
0025 0062   10F9            JMP   LOOP
0026 0064   045B   RET      B     *R11
0027                        END
```

A	0004	C	0008	CLEAR	0050	D	0000
DISP	0034	LOOP	0056	R0	0000	R1	0001
R10	000A	R11	000B	R12	000C	R13	000D
R14	000E	R15	000F	R2	0002	R3	0003
R4	0004	R5	0005	R6	0006	R7	0007
R8	0008	R9	0009	RET	0064	RT	0002
D START	0028	STATUS	837C	STOP	0044	E VMBW	0042
E VSBW	0058						

0000 ERRORS

```
0001          DEF   START
0002          REF   VSBW,VMBW
0003  STATUS  EQU   >837C
0004  D       DATA  >2000
0005  RT      DATA  >0000
0006  A       TEXT  'NOME'
0007  C       BSS   >20
0008  START   MOV   R11,@RT
0009          LWPI  C
0010          BL    @CLEAR
0011  DISP    LI    R0,299
0012          LI    R1,A
0013          LI    R2,4
0014          BLWP  @VMBW
0015  STOP    MOV   @RT,R11
0016          CLR   @STATUS
0017          DECT  R11
0018          RT
0019  CLEAR   CLR   R0
0020          LI    R1,D
0021  LOOP    BLWP  @VSBW
0022          CI    R0,767
0023          JEQ   RET
0024          INC   R0
0025          JMP   LOOP
0026  RET     B     *R11
0027          END
```

== SPIEGAZIONE DEL PROGRAMMA LINEA PER LINEA ! ==

LINEA 0001 **DEF START**

L'istruzione **DEF** (DEFinisci) e' una direttiva Assembler, non ha nessun contrasto con l'esecuzione logica del programma, essa ha solo il compito di mettere il nome dato al programma, nel nostro caso **"START"**, dentro un area del computer conosciuta come **TAVOLA REF/DEF**. In questo posto vengono conservati tutti i nomi dei programmi che saranno caricati, la direttiva **DEF** ti permette di mettere in esecuzione il tuo programma chiamandolo per nome ("START"), in pratica come il **RUN** del Basic.

LINEA 0002 **REF VSBW,VMBW**

La direttiva **REF** dice all'Assembler che voi intendete usare alcuni programmi speciali residenti, essa assicura che quando il vostro programma e' caricato, queste routines saranno disponibili. **VSBW** sara' reso uguale all'indirizzo della routine in **VDP RAM** per scrivere un singolo Byte, anche per **VMBW** accadra' la stessa cosa, ma con la differenza che scrivera' piu' Byte.

LINEA 0003 **STATUS EQU>837C**

Questa e' l'istruzione **EQUate** (eguaglia), essa e' anche una direttiva, la direttiva **EQU** associa l'etichetta data (STATUS) con l'indirizzo **>837C** che in pratica sarebbe lo Status Byte.

LINEA 0004 **D DATA >20**
0005 **RT DATA >0000**

Queste linee usano le direttive **DATA**, asse sono direttive usate per inizializzare delle Word di memoria per qualche valore, se una etichetta e' inclusa quest'ultima viene associata con l'indirizzo iniziale della Word. In pratica le istruzioni **DATA** hanno eseguito un'assegnazione di un valore come in Basic avviene con la istruzione **LET**, prendiamo le istruzioni del programma nella linea 0004 la lettera A ha preso un valore HEX di >20 (o decimale 32) invece per **RT** il valore e HEX >0000 (in decimale 0).

LINEA 0006 **A TEXT 'NOME'**

La direttiva **TEXT** ha il compito di assegnare alla etichetta A la parola **NOME** la quale deve essere racchiusa dai segni di virgolette semplici (') esse non faranno parte di cio' che sara' visualizzato.

LINEA 0007 **C BSS >20**

La direttiva **BSS** (Block Starting with Simbol) riserva dei blocchi di memoria senza alcuna inizializzazione, queste linee saranno unaste come Workspace e saranno parte del progrmma stesso, in questo caso alla lettera **C** sono attribuiti 32 Bytes (HEX >20).

LINEA 0008 **START MOV R11, RT**

Questa linea e' l'inizio vero e proprio del programma, qui c'e' l'etichetta **START** che e' stata definita alla linea 0001, il primo obbligo e' quello di salvare l'indirizzo del programma. Quando il vostro programma Assembler inizia l'esecuzione, l'indirizzo al quale esso dovrebbe tornare una volta eseguito e' in **R11**, questo indirizzo e' essenziale per il corretto funzionamento del programma stesso. In questo caso noi dovremo salvare questo indirizzo, poiche' **R11** sara' usato altrove nel programma. La Word di memoria **RT** era stata messa via per questo, con l'istruzione **MOV** si muovera' il valore contenuto da **R11** in **RT**, supponendo che **R11** conteneva il valore >3126, dopo **MOV RT** conterra' anchesso questo valore.

LINEA 0009 **LWPI C**

La direttiva **LWPI** (Load Workspace Pointer Immediate) carica immediatamente il puntatore dello spazio di lavoro, Voi avete bisogno di stabilire un'area alternativa dei Registri Workspace per l'uso di routine speciali che sono necessarie al programma. L'effetto di questa istruzione e' di puntare all'indirizzo del blocco di memoria che era stato riservato nella linea 0007.

LINEA 0010 **BL CLEAR**

Invece di chiamare una routine residente per pulire lo schermo, questo programma ha una sua routine codificata, la istruzione **BL** (Branch and Link) ha il compito di diramarsi od allacciarsi alla etichetta **CLEAR**, in questo punto **R11** contiene l'indirizzo della istruzione successiva che e' **DISP**. In pratica e' come le istruzioni **CALL** dell'Ext. Basic, esse chiamano dei sub programmi per nome e non per numero di linea, per ulteriori informazioni consultare il manuale X/B.

LINEA 0011 **DISP LI R0,299**

In questa linea l'istruzione **LI** (Load Immediate) carica immediatamente il valore decimale **299** in **R0**, il numero **299** e' la posizione dello schermo dal quale dobbiamo stampare il nome, per una migliore comprensione consulta la tabella dello schermo allegata. L'etichetta **DISP** in questo caso serve solo per ricordarsi cio' che si eseguira' nelle linee subito sucessive.

LINEA 0012 **LI R1,A**

Come sopra ma in questo caso in **R1** viene caricato la stringa contenuta in **A**, cioe' il 'NOME'.

LINEA 0013 **LI R2,4**

Come nella linea 0011, **LI** carica in **R2** il valore **4** che e' la lunghezza della stringa **A** ('NOME').

LINEA 0014 **BLWP VMBW**

L'istruzione **VMBW** (Vdp Muliple Byte Write) scrive il contenuto di **R1** nella posizione contenuta in **R0** per una lunghezza contenuta in **R2**, questa istruzione esegue la visualizzazione mediante i Registri che contengono tutti i dati.

LINEA 0015 **STOP MOV RT,R11**

In pratica questo e' il punto dove il programma termina, quindi noi ora dobbiamo ritornare da dove eravamo partiti, in questo caso muoveremo il valore precedentemente salvato in **RT** nel Registro **R11**.

LINEA 0016 **CLR STATUS**

L'istruzione **CLR** (CLeaR) pulisce il contenuto della matrice **STATUS**.

LINEA 0017 **DECT R11**

A causa della eccessiva velocita' di esecuzione del programma in questa linea l'istruzione **DECT** decrementa di 2 il valore contenuto in **R11**, in questo modo il programma si congelera' fino a quando si premera' FCTN QUIT, se eliminate questa istruzione vedrete il programma terminare in una velocita' pazzesca.

LINEA 0018 **RT**

L'istruzione **RT** (ReTurn) ha il compito, nel caso non ci fosse la istruzione nella linea 0017, di ritornare dove eravamo quando abbiamo chiamato il programma.

LINEA 0019 **CLEAR CLR,R0**

L'istrizione **CLR** (CLeaR) pulisce il Registro **R0**.

LINEA 0020 **LI R1,D**

L'istruzione **LI** (Load Immediate) carica immediatamente il valore contenuto in **D**, in questo caso e' il codice ASC. dello spazio >20 (DEC. 32), in **R1**, ora **R1** conterra' >20.

LINEA 0021 **LOOP BLWP VSBW**

L'istruzione **VSBW** (Vdp Single Byte Write) scrive un singolo Byte per volta, in pratica come **VMBW** con la sola differenza che essa richiede solo due Registri **R0** ed **R1** nei quali dovrete mettere cio' che desiderate visualizzare. In questo caso **R0** contiene 0 che e' la prima posizione in alto a sinistra dello schermo ed **R1** il codice Hex. >20 del carattere spazio.

LINEA 0022 **CI R0,767**

L'istruzione **CI** (Compara Immendiate) confronta immediatamente il valore contenuto in **R0** con 767, essa influisce sui Bit di Stato nel caso la comparazione e' vera. Per una migliore comprensione consultare le tabelle dei Registri allegate.

LINEA 0023 **JEQ RET**

L'istruzione **JEQ** effettua un controllo sui Registri di stato e se la condizione e' vera salta all'etichetta **RT**.

LINEA 0024 **INC,R0**

Questa istruzione incrementa di 1 il Registro **R0**, in pratica R0=R0+1.

LINEA 0025 **JMP LOOP**

L'istruzione **JMP** e' un salto incondizionato alla etichetta **LOOP**.

LINEA 0026 **RET B *R11**

L'istruzione **B** (Branch) e' simile alla istruzione GOTO del Basic, essa e' una diramazione incondizionata all'indirizzo contenuto in **R11**.

LINEA 0027 **END**

Questa e' l'ultima istruzione del programma e deve essere scritta obbligatoriamente, altrimenti l'assemblatore segnalera' un'errore.